

Project Overview

OpenTelemetry is an open source software that provides a framework for observability by enabling the instrumentation, generation, collection, and export of telemetry data - metrics, logs, and traces.

Has 75 repositories that have a collection of APIs, SDKs and other tools.

Goals and Milestones

Milestone 1: Download, build, and test out the existing OpenTelemetry Project codebase. Understand the codebase and the contributions required to achieve the final goal. Talk to the mentor about any issues and the OpenTelemetry maintainers if necessary.

Milestone 2: Create the necessary PRs for achieving the goal of the project. Resolve at least 2 issues in the opentelemetry-python repository.

Open Source Outcomes

- Made the code changes required to elevate community support for linux/s390x for the OpenTelemetry Collector from Tier 3 to Tier 2.
- Compiled and ran the OpenTelemetry Collector on IBM z/OS and created documentation to reproduce the same.
- Implemented the Events API
- Created PRs to resolve 7 issues in the OpenTelemetry Python repository.

Students: Soumyadeep Mahapatra
Mentor: Rüdiger Schulze

Elevating community support for linux/s390x for OT Collector

- Required integrating the s390x runner in the CI/CD pipeline.
- However, there is no GitHub hosted runner for s390x, but there is an option of using a self-hosted runner.
- Using a self-hosted runner has security concerns.
- GitHub hosted runner is a clean isolated VM created when a job starts and destroyed when it ends which is not the case with self-hosted runners which creates an issue.
- Working to resolve ongoing issue in the repo regarding security problems with self-hosted runner. Pushed code changes in SIG Mainframe repo till the issue is resolved.

7 Resolved Issues

- Implemented the Events API:** There is specification document for the events API but it was not implemented in a lot of repos including the OT Python repo.
- _encode_events assumes events.attributes.dropped exists:** attributes in events object was of type `Optional[Mapping[str,AttributeValue]]` which did not guarantee that it had a dropped attr.
- OTLP exporter is encoding invalid span/trace IDs in the logs:** When there is no active trace, the exporter was including span/trace IDs in the protobuf messages which should not have been happening.
- Optional scope attribute for tracer/logger creation:** The specification sheet required that when tracers or loggers are created, users should be able to add optional attributes to it.
- Confusing error when span name is not a string:** When spans are created and the name passed as a parameter is not a string, it gave "TypeError: bad argument type is not a built-in operation" which does not help the user much in identifying their mistake.
- Resolved 2 more issues in addition to the ones above.

Highlights and Accomplishments

Compiling and running OT Collector on z/OS

- Involved testing various go versions and modifying the configuration of the Collector.
- "Make" available on z/OS didn't work as expected so had to use GNU Make from z/OS Open Tools to compile the collector.
- No implementation for metric collection on z/OS.

- Have to turn off metrics in the Collector configuration file to run the Collector on z/OS.

```
unittest-matrix:
  strategy:
    matrix:
      runner: [ubuntu-latest, actuated-arm64-4cpu-4gb, <linux_s390x-runner>]
      go-version: ["~1.22", "~1.21.8"] # 1.20 needs quotes otherwise it's interpreted as 1.2
    runs-on: ${matrix.runner}
    needs: [setup-environment]
    steps:
      - name: Set up arkade
        if: ${matrix.runner == 'actuated-arm64-4cpu-4gb'}
        uses: alexellis/setup-arkade@v3
      - name: Install vmmeter
        if: ${matrix.runner == 'actuated-arm64-4cpu-4gb'}
        run: |
          sudo -E arkade oci install ghcr.io/openfaasid/vmmeter:latest --path /usr/local/bin/
      - name: Run vmmeter
        if: ${matrix.runner == 'actuated-arm64-4cpu-4gb'}
        uses: self-actuated/vmmeter-action@v1
      - name: Checkout Repo
        uses: actions/checkout@04ffde65f46336ab88eb53be898477a3936bae11 # v4.1.1
```

Build OpenTelemetry Collector on z/OS

You can build the collector on z/OS from the source by following the steps below:

Requirements

To build the collector, you need GNU make and go v1.21.1. You can find and install them using the links below:

- GNU Make
- go 1.21.1

Compiling the collector

Compile the opentelemetry collector repository using the following commands:

```
git clone https://github.com/open-telemetry/opentelemetry-collector.git
cd opentelemetry-collector
make install-tools
make install
```

This will generate the collector executable in the `bin` directory.

Configuring the collector

Configuring the collector does not run without turning off metric collection. To disable metric collection, set the `metrics.level` to `disabled`. An example config file is shown below:

```
metrics:
  level: disabled
  processors:
    - type: drop
      log_level: detailed
  service:
    log_level: detailed
    log_file: /tmp/ot.log
    log_max_size: 10000
    log_max_time: 10000
    log_rotate_time: 10000
  telerometer:
    metrics:
      level: disabled
      processor:
        type: drop
        log_level: detailed
  reporter:
    type: file
    log_level: detailed
    log_file: /tmp/ot-reporter.log
    log_max_size: 10000
    log_max_time: 10000
    log_rotate_time: 10000
```

Running the collector

Run the collector using the executable and the config file:

```
./bin/otcollector --configPath=src/config.yaml
```

Replace `*` with the path to your config file.

```
CHANGELOG.md
src/opentelemetry-api
src/opentelemetry
  __events
    __init__.py
  environment_variables
    __init__.py
  tests/events
    test_event.py
    test_event_logger_provider.py
    test_proxy_event.py
  tests/opentelemetry-test-utils/src...
  globals_test.py

  + class Event(LogRecord):
  21 +     def __init__(_
  22         self,
  23         name: str,
  24         timestamp: Optional[int] = None,
  25         trace_id: Optional[int] = None,
  26         span_id: Optional[int] = None,
  27         version: Optional[int] = None,
  28         payload: Optional[Any] = None,
  29         severity_number: Optional[SeverityNumber] = None,
  30         attributes: Optional[Attributes] = None,
  31     ):
  32         super().__init__(
  33             name=name,
  34             timestamp=timestamp,
  35             trace_id=trace_id,
  36             span_id=span_id,
  37             trace_flags=trace_flags,
  38             body_payload=payload,
  39             severity_number=severity_number,
  40             attributes=attributes,
  41         )
  42         self.name = name
  43
  44 + class EventLogger(ABC):
  45 +     def __init__(_
  46         self,
  47         name: str,
  48         version: Optional[str] = None,
  49         schema_url: Optional[str] = None,
  50         attributes: Optional[Attributes] = None,
  51     ):
  52         super().__init__(
  53             name=name,
  54             version=version,
  55             schema_url=schema_url,
  56             attributes=attributes
  57         )
  58         @abstractmethod
  59         def emit(self, event: "Event") -> None:
  60             """Emits a :class:`Event` representing an event."""
  61
  62 + class NoOpEventLogger(EventLogger):
  63 +     def emit(self, event: Event) -> None:
  64             pass
  65
  66 + class ProxyEventLogger(EventLogger):
  67 +     def __init__(_
  68         self,
  69         name: str,
  70         version: Optional[str] = None,
  71         schema_url: Optional[str] = None,
  72         attributes: Optional[Attributes] = None,
  73     ):
  74         super().__init__(
  75             name=name,
  76             version=version,
  77             schema_url=schema_url,
  78             attributes=attributes,
  79         )
```

Future Work

- Compiling OT Collector Contrib on z/OS - Tried doing this but there were many issues and didn't have time to resolve it.
- Elevate community support for OT Collector for linux/s390x to Tier 1 - Involves adding performance tests and some end-to-end tests.